08

Resolviendo conflictos

Transcripción

[00:00] Hola a todos y todas. Bienvenidos y bienvenidas una vez más a este curso de Git. Vimos un caso interesante donde Bruno corrigió un bug pero esa tarea también fue ejecutada por Ana. Entonces, los dos modificaron la misma línea de código. ¿Entonces qué va a ocurrir si intento juntar ese trabajo?

[00:18] Puedo elegir si voy a hacer merge o rebase, no hay mucha diferencia. En este caso voy a elegir hacer un merge. Estando acá con Ana, bien, voy a hacer un git merge lista. ¿Qué es lo que ocurre? Miren lo que nos está diciendo acá. Que hay un conflicto. Entonces ese merge automático falló. Dice primero que hay que corregir esos conflictos y después de eso podemos continuar.

[00:50] Entonces vamos a abrir el archivo y vean lo que tenemos acá. Fíjense, tenemos lo que está entre la palabra head y los signos de igual son lo que tenemos en nuestra rama master actualmente. Y lo que está entre los signos igual y la palabra lista es lo que está en la branch lista, que es lo que queremos mergear.

[01:17] Entonces acá en Visual Studio Code puede ser que ustedes, estamos viendo con colorcitos, todo muy lindo, pero si están utilizando tal vez otro IDE es posible que no vean todos esos colores. Entonces solamente vamos a enfocarnos en estas palabras, estas partes. Entonces me está mostrando exactamente la diferencia entre esos dos commits. Lo único que tengo que hacer para corregir ese conflicto es eliminar los datos que no quiero. Eliminar lo que no necesito.

[01:45] Dejar solo la línea en la versión correcta. Estoy viendo por ejemplo que quedó mejor la forma que está en la branch Bruno, que está en lo que hizo Bruno, en esa línea, sin el curso en el comienzo. Entonces vamos a aceptar la modificación que hizo él. Voy a eliminar lo que estaba hecho acá en la branch de Ana, voy a eliminar eso, voy a sacar la palabra head y dejo todo así. Vamos a hacer un "Ctrl + S" para guardar.

[02:13] Entonces si ahora voy acá, a donde está el proyecto de Ana voy a hacer un git status. Enter. Vemos que nos está diciendo que hay dos archivos han sido modificados, que es el index de una branch de Ana y de Bruno. Y entonces ahora que ya corregí ese problema voy a hacer un git add index.html Y así informo a Git que ya corregí lo que tenía que corregir.

[02:42] Entonces a partir de que corregí y agregué ese index puedo hacer un git commit. Enter. Git va a saber que terminé de corregir los conflictos y va a hacer un commit de merge. Entonces acá estando en esta pantalla puedo hacer :x para salir damos enter y con esto guardé ese mensaje y actualicé mi branch master.

[03:12] Si ejecuto ahora en git log, espacio, --graphs, enter tengo el merge en mi branch. Genial. Ahora puedo hacer un git push servidorlocal, master. Vamos a apretar Q y vamos a hacer git push servidorlocal master para enviar los datos. Y ahora imaginen que Bruno está trabajando en el proyecto corrigiendo nuestro curso de Vagrant.

[03:47] Entonces, por ejemplo, dentro del archivo de Bruno, realmente el curso de Vagrant en realidad se llamaba Vagrant: Gerenciando máquinas virtuales. Con esto guardamos el archivo y voy a loguearme como Bruno. Vean que estoy acá como Bruno y voy a hacer un git status.

[04:17] Bien. Vemos que está modificado git add punto. Git commit -m "corrigiendo curso de Vagrant". Damos un enter allí y mientras envío esa información voy a hacer un git push servidorlocal master y miren lo que

ocurre. Mientras Bruno estaba trabajando, Ana envió esa información, ese merge que hicimos, al servidor en la rama master antes que Bruno.

[04:55] Necesito entonces antes de enviar cualquier modificación mía de Bruno, tengo que asegurarme que estoy trabajando con la versión más nueva del código. Entonces antes de enviar tengo que traer ese código al servidor. Entonces voy a hacer un git pull servidorlocal, master, damos enter y ahora sí damos un :x y ahora sí con los datos ya en nuestro proyecto ahora puedo por ejemplo verificar y vemos que tenemos nuestro curso Vagrant.

[05:28] Están todos los datos ya que habían sido modificados y ahora sí puedo enviar esa modificación. Entonces voy a hacer acá como Bruno, git push servidorlocal master, damos Enter y ahora sí ya conseguí enviar los datos. Siempre que vayamos a comenzar a desarrollar un proyecto y antes de enviar los datos de ese desarrollo que hagamos, sé que tengo que verificar si existe alguna modificación en el repositorio.

[06:00] Entonces con Ana, antes enviar alguna modificación nueva, ella sabe que necesita hacer un git pull del servidor local para estar segura que no hay ninguna modificación allá en nuestro repositorio remoto. Entonces, vamos a hacer eso, vamos a cambiar para Ana, vamos a hacer un git pull servidorlocal master. Enter.

[06:23] Como vemos, ya nos trajo los datos nuevos que tenían allá en el repositorio. Entonces para evitar conflictos Git ya nos previene de esos conflictos, pero cuando ocurren, ya vimos que es bastante fácil de resolverlos. Bueno, ya aprendimos cómo trabajar con repositorios remotos, cómo trabajar en equipos, cómo trabajar con branches separadas y cómo unir el trabajo de branches diferentes.

[06:31] Quiero dejar una observación: existen estrategias bien específicas de cuándo crear una branch, cómo crear una branch, necesito crear una branch

para cada funcionalidad nueva o no. En este caso no voy entrar en detalles sobre este tema porque no es el foco realmente de este entrenamiento.

[07:06] Quizás en entrenamientos futuros entremos en detalle de esas estrategias de branches, pero sabemos que branches son líneas de desarrollo y aprendimos a crearlas, unir trabajos de branches diferentes, sean a través de merge o rebase y aprendimos a resolver conflictos.

[07:23] Ahora que tenemos esa información, ¿cómo será que puedo navegar por el histórico de ese proyecto? ¿Cómo puedo deshacer una modificación, desistir de una modificación? ¿Cómo puedo ver una modificación que hice hace un tiempo? ¿Cómo puedo navegar en sí en mi proyecto? Bueno, vamos a hablar sobre eso en el próximo capítulo.